Proceedings, 4th International Workshop on Economics-Driven Software Engineering Research,
Orlando, Florida, May 21 2002, pp. 10-14.

A Risk-Centric Model for Value Maximization

Martin S. Feather
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
+1 818 354 1194
Martin.S.Feather@Jpl.Nasa.Gov

Steven L. Cornford
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
+1 818 353 5365
Steven.L.Cornford@Jpl.Nasa.Gov

Julia Dunphy
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
+1 818 354 1701
Julia.Dunphy@Jpl.Nasa.Gov

ABSTRACT

We describe an approach that makes value maximization possible during the earliest phases of software development. In this context it is challenging to determine which of many possible concerns are most critical, and therefore the basis for decision-making. Yet, decisions made in these early stages have the greatest opportunity to influence the future course of development.

Our approach is based on the use of a risk-centric model that we have been developing and applying to assessment and planning of systems development (not necessarily software systems). This model quantitatively links requirements to risks, and risks to mitigations. Value is measured in terms of requirements attainment, while cost is measured in terms of the resources needed to apply the chosen risk mitigations. Risk is the connection between the two.

We have constructed custom tool support for applying this model during group sessions, to gather information on the fly from experts, and to aid them in decision-making based on the combination of the information they have provided. The tool support makes feasible the accommodation of the relatively large number and wide variety of concerns. In actual applications it has proven useful in guiding experts to make their most critical early phase decisions.

Categories and Subject Descriptors

B.m [**Miscellaneous**]: Design management. C.4 [**Performance of Systems**]: Design studies, Modeling techniques. D.2.1 [**Software Engineering**]: Requirements/Specifications - *Elicitation methods*. D.2.9 [**Software Engineering**]: Management – *Cost estimation, life cycle, Software process models, Software quality assurance*.

General Terms

Management, Measurement, Performance, Design, Economics.

Keywords

Risk, risk management, risk mitigation, risk assessment, cost-benefit analysis, requirements, software assurance.

1. INTRODUCTION: A QUANTITATIVE RISK-CENTRIC MODEL OF COST AND VALUE

At NASA we have been developing and applying our risk management framework, "Defect Detection and Prevention" (DDP), for several years. DDP is a process for which we have custom-built software support. We have reported on DDP in other forums – for an overview see [1].

DDP is intended for use in the early phases of system development. The motivation is that these early phases have the maximum opportunity to influence the development to follow. They are characterized by numerous concerns that span both product and process, involve multiple stakeholders (customers, developers, maintainers and users), but are lacking in well-worked out designs. Our approach is to convene experts who represent all the stakeholder positions, elicit from them the knowledge that is crucial to planning the subsequent development, and aid them in identifying and making critical decisions based on this knowledge. Note that our aim is to support experts in their decision making, not to replace them. The involvement of those experts is retained throughout the process.

DDP deals with three key concepts: requirements, risks and risk mitigations (in some of the papers we have published, risks are referred to as "failure modes", and mitigations as "PACTs"). Risks are quantitatively related to requirements, to indicate how much each risk, should it occur, impacts each requirement. Mitigations are quantitatively related to risks, to indicate how much of a risk-reducing effect the mitigation, should it be applied, has on the risk.

The *value* of a DDP model is measured in terms of requirements attainment. Requirements are individually "weighted", to reflect their relative importance. Risks adversely impact requirements attainment. The *cost* of a DDP model is measured in terms of the resource costs of the mitigations selected. Mitigations reduce risks, and so lead to increased attainment of requirements, but incur costs. The primary purpose of DDP is to facilitate judicious selection of a set of mitigations, thus attaining requirements in a value-maximizing manner. If requirements are valued in the same

units as the resource costs of mitigations, then it is possible to directly trade requirements for cost.

2. DDP IN USE

In our NASA applications of DDP, the systems being developed are spacecraft, or components of spacecraft. Hence the stakeholders include the mission scientists whose science needs become requirements, the spacecraft experts who understand the different disciplines involved in the system's operation (e.g., power, navigation, communications), the engineers who understand the challenges in designing, developing, integrating and testing the spacecraft's components, and the mission operators who will control the spacecraft during through its use.

We use these sessions to (1) gather the requirements, (2) gather the risks (a.k.a. "failure modes"), and how much they impact the requirements, (3) gather the mitigations, and how much they effectively reduce risks, and (4) select mitigations (and sometimes discard requirements, when it becomes clear that a requirement is not worth the cost it takes to reduce the risks impacting it).

Underpinning DDP is a simple quantitative model of cost and value. We use risk as the intermediary between requirements (whose attainment provides the value) and mitigations (whose selection incurs cost). In the subsections that follow we describe this model, emphasizing its quantitative nature and how it is applied in practice.

2.1 Requirements

We gather the wide range of requirements relevant to the study at hand. These may include both requirements imposed on the system to be developed (e.g., memory usage, throughput, accuracy) and requirements on the development process itself (e.g., schedule, budget, programming language, programming development environment). In typical DDP applications experts have listed 30 – 100 requirements,. These same experts assign weights to the requirements to indicate their value. Note that these expert-assigned weights are in units of the experts' choosing, and are *cardinal* (not ordinal) in nature. They are additive, i.e., the value of a set of requirements is the sum of the values of the requirements in that set. This quantitative treatment allows us to compare the value of various combinations of requirements.

2.2 Risks

We treat risk in its most general sense – anything that, should it occur, will cause loss of requirements. Brainstorming all the ways that things could go wrong is an important part of DDP sessions. In DDP applications we have gathered from 30 – 200 risks.

Risks are quantitatively linked to requirements. For each risk x requirement pair, the experts provide an estimate of the "impact" of the risk on the requirement, which we define as *the proportion of the requirement that would be lost were that risk to occur*. In order for the overall process to have value, it is not necessary for these estimates to have great precision. In practice we have found that the gradations 1, 0.9, 0.3, 0.1, 0 are sufficient for most situations. 1 means certain loss (i.e., if the risk occurs, the requirement will be totally lost); 0.9 means high impact – technically, we interpret the figure to mean the proportion of the requirement that will be lost should the risk occur; etc. In practice, many of the risk x requirement pairs have an impact of 0. Nevertheless, in real applications hundreds, sometimes thousands, of risk x requirement pairs are assigned non-zero values.

We use disagreement to drive the need for more detail. When two or more experts disagree about an impact value, it is usually because they are referring to different circumstances. We accommodate their respective positions by refining the requirement and/or the risk into subcases, as appropriate.

Our DDP model assumes that risks combine additively. For example, if two different risks both impact the same requirement, then their combined impact is the sum of their individual impacts. This simplistic model is easy to understand and work with, and has been used in our DDP applications to date. We are in the process of introducing logical concepts to the risks, along the lines of the logical fault trees (e.g., AND and OR nodes) used in probabilistic risk assessment. These will enable to the model to accommodate more of the logical structure that emerges as the early details of a design becomes known.

The total impact that risks have on a given requirement is computed as:

RisksImpactOnRequirement(q) ≡
Weight(q) *
(Σ (r ∈ risks) : Impact(r, q) * APrioriLikelihood(r))

where

Weight(q) is the user-assigned weight of requirement q,
Impact(r, q) is the proportion of requirement q lost should risk r occur, and
APrioriLikelihood(r) is the a priori likelihood of risk r.

Hence, taking risks into account, the amount of attainment of requirement r is:

AttainmentOfRequirement(q) ≡
Weight(q) *
(1 – (Σ (r ∈ risks) : Impact(r, q) * APrioriLikelihood(r)))

However, it is possible for the sum of risks' impacts on a requirement to exceed 1, in which case the value computed above would become negative. So in practice we compute the attainment of requirement r as the maximum of this and zero, thus:

AttainmentOfRequirement(q) ≡
Weight(q) *
Max(0, (1 – (Σ (r ∈ risks) : Impact(r, q)
APrioriLikelihood(r))))

2.3 Mitigations

We treat mitigation in its most general sense – anything that, should we choose to do it, will decrease the likelihood of risks and/or their severity on requirements. Once again, they are numerous; we have seen 30 – 170 in applications to date.

Mitigations are quantitatively linked to risks. For each risk x mitigation pair, the experts provide an estimate of *the proportion by which the risk would be reduced were that mitigation to be applied*. Again, coarse estimates of effectiveness suffice for our purposes. The same gradations as for risk impacts, but with 0.9 in place of the 1 (i.e., 0.99, 0.9, 0.3, 0.1, 0) are sufficient for most situations. We prefer 0.99 in place of 1 on the grounds that there is very rarely a "perfect" mitigation.

Our DDP model assumes that mitigations combine as follows: suppose one mitigation has effectiveness E1 on a risk (i.e.,

removes proportion E1 of that risk), and another mitigation has effectiveness E2 on that same risk, then together they have a combined effectiveness of (1 – (1 – E1)*(1 – E2)). Roughly speaking, they act like filters in series: proportion (1 – E1) of risks make it through the first filter, and of these, (1 – E2) of them will make it through the second filter. Again, this is a simple model that is easy to understand and work with, and that appears to match our experts' intuition of how various risk mitigation techniques combine.

The total effectiveness that mitigations have on a given risk is computed as:

MitigationOfRisk(r) ≡
1 - (Π (m ∈ mitigations) : (1 - Effect(m, r)))

where

Effect(m, r) is the effect of mitigation m at reducing risk r.

Thus taking mitigations into account,

MitigatedRisksImpactOnRequirement(q) ≡
Weight(q) *
(Σ (r ∈ risks) : Impact(r, q) * APrioriLikelihood(r)
* (1 – MitigationOfRisk(r)))

Performing a mitigation incurs the resource costs associated with that mitigation, which could be measured in dollars, schedule, availability of test harnesses, etc. Mitigations can be associated with major development times or phases at which they are performed, allowing the costing to be tracked with respect to those phases.

2.4 Decision-Making

In the final step of a DDP application, the experts scrutinize the combination of the information they have provided, and use it to guide their decision-making.

2.4.1 Support for Users' Decision Making

Although DDP's models are relatively simplistic, their scale justifies the need for DDP's automated calculations. DDP is able to compute the value (requirements attainment) and cost (mitigations and repairs) for a given selection of mitigations. DDP uses cogent visualizations to present different views of the risk model, and its consequences, to those experts.

An example such visualization is shown in Fig. 1. This is data taken from an actual DDP application. It shows a bar chart of requirements, where a separate bar represents each requirement. The color-coded information conveys the current status of requirements attainment, including for each requirement the user-assigned weight (indicated by the blue line segments), the extent to which it is currently at risk (indicated by the red portion of bars) and the extent to which the currently selected mitigations have reduced risk (indicated by the green portion of bars). Bars can be sorted in various ways. The figure shows them sorted in descending order of the user-assigned weights. Another useful way to sort them is in descending order of remaining risk.

Similar DDP generated charts give experts insight into the risks, allowing them to see what their most pressing problems are, and into the mitigations, allowing them to see what their most effective solutions are. There are also several ways to view the impact relationships (between risks and requirements) and the effect relationships (between mitigations and risks).

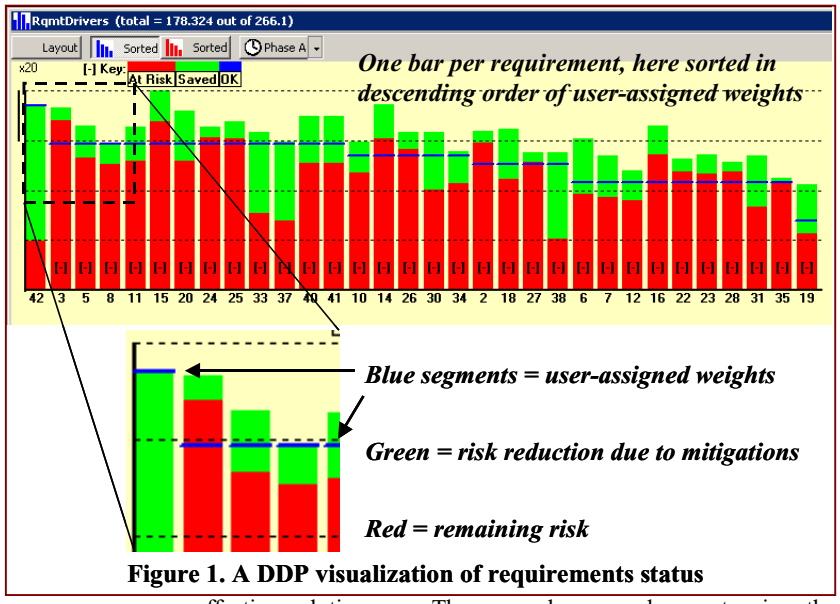

Figure 1. A DDP visualization of requirements status

2.4.2 Range of Decisions

In applications to date, the experts themselves have been able to use DDP to help them make a range of decisions. We have seen cases of using DDP to:

Value maximization: selection of the mitigations that minimize the cost needed to achieve the desired value (requirements attainment), or, for a given cost, achieve the maximum possible value. This has the obvious benefit of cost-effectiveness. It has the additional benefit of making clear the contribution(s) of the mitigations at reducing risk. This increases the motivation for performing mitigations - they become seen as beneficial, rather than simply hurdles to overcome. Finally, it gives an indication as to which kinds of risks the mitigations are being relied upon to prevent, which can be useful guidance to the people performing the mitigations.

Requirements triage: identification requirements proving to be the most problematic (costly) to attain. In one application, the focus upon one such requirement led to its clarification, the net result of which was considerable savings in work not required.

Assessment and planning: make a thorough assessment of the viability of novel advanced technology, and emerge with a well-structured plan for maturing that technology towards use.

3. SOFTWARE ASSURANCE PLANNING

Most of our experience with the use of DDP has been on assessment and planning of advanced spacecraft technologies with a significant hardware component. Relatively fewer DDP studies have had software as their primary concern. Nevertheless, we are confident that DDP has applicability to hardware, software and hardware-software combinations, since requirements, risks and mitigations are fundamental to all three areas. In this section we focus on the work we have been doing to make DDP applicable to, specifically, software assurance planning.

3.1 Refinements of DDP's Model

We have refined DDP's cost-benefit model to encompass some of the phenomena that arise in costing of, especially, software assurance activities.

3.1.1 Risk Likelihood vs. Risk Severity

It is common practice to distinguish between likelihood and severity in the calculation of risk. DDP already had in place the means to assess the (unmitigated) severity of risks, as the sum of impacts on weighted requirements. By refining mitigations into those that reduce likelihood vs. those that decrease severity, we are able to continue to distinguish between likelihood and severity throughout the calculations. Examples of risk likelihood reductions include training, following coding standards, use of design patterns. Also, analysis, testing, etc., falls into this category, as we explain in the next subsection. Examples of risk severity reductions include use of bounds checking on inputs, and error handling schemes (fault protection for software).Fig. 2 shows DDP's "risk region" visualization that makes use of this distinction. It plots risks (one small black square for each risk) in two dimensions, likelihood and severity. The axes of the chart are log scale, so the diagonal boundaries between the three colored regions are in fact lines of constant risk. The three regions subdivide risks in to "high" - top-right corner, "medium" - center region, and "low" - bottom left corner. In fact, there is a fourth category, "tiny", risks that are so small they do not appear on the chart at all (because the chart axes are log scales, zero risk would be a point at minus infinity).

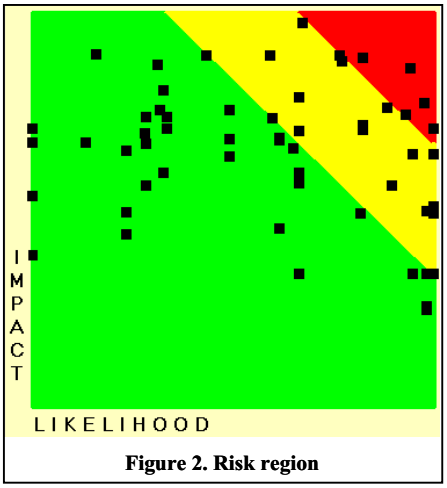

Figure 2. Risk region

3.1.2 Risk Prevention vs. Detection (and Repair)

We further subdivide the risk likelihood reducing mitigations into those that prevent risks from arising in the first place, and those that detect the presence of risks. The latter are assumed to be coupled with the appropriate action to repair the defects so detected. For example, unit testing is applied to detect coding problems, the assumption being that bugs uncovered during unit testing will be fixed. The net result is a reduction in the likelihood of risks remaining in the product through to the next phase of development (and ultimately through to release).

Having made this distinction, we can now subdivide the associated costs between the cost of detection and the cost of repair. The latter is especially important, since it escalates dramatically the later in the lifecycle a problem is detected. For example, the often-quoted statistics on how the cost of correcting a requirements problem escalates through design, coding, testing and release. In the DDP model we associate a cost with the detection-style mitigation (whatever resources are needed to perform that mitigation), and a separate cost with the risk (whatever resources are needed to repair that risk should it be detected). The latter is time-dependant, through which we can incorporate the escalation of costs in later development phases.

We can use DDP to demonstrate quantitatively the net benefits of early-phase mitigations, because they prevent or detect problems early, thus saving the much greater costs associated with repairing them late. For more on this, see [2].

3.1.3 Risk Introduction

We allow for the possibility that the use of a mitigation can introduce risk. For example, code added to enhance testability might introduce new bugs (or its removal might change the real-time behavior). The motivation for this derives more from the application of DDP to hardware studies, where tests have a physical manifestation.

In the software arena, a related phenomenon is that of code added to correct one problem introducing new problems. This possibility of repairs introducing risk (rather than mitigations) is *not* yet in our model, but it something we plan to add given its relevance to software development.

3.2 Software Assurance Specific Information

We have worked on pre-populating DDP with information specific to software assurance planning. Known software development risks become DDP risks, and known software assurance practices become DDP mitigations.

- A taxonomy prepared by the SEI serves as our source of candidate software development risks. DDP users can remove risks that are irrelevant (e.g., if a risk concerns subcontracting, and the project under scrutiny has no subcontracting, then it would be appropriate to remove that risk).
- A listing of software assurance activities that are part of another NASA tool, Ask Pete (see section 3.3.1), serves as our source of candidate mitigations. Again, DDP users can remove mitigations that are impractical (e.g., because the personnel with the skills to apply them are unavailable).
- We made estimates of the effectiveness of each of the mitigations are reducing each of the risks. While we would prefer to use known effectiveness figures based on experience, this data is generally lacking. In its absence, we work with estimates. In the software realm, we look to groups such as the CeBASE consortium http://www.cebase.org to gather such data.

Users can add to, remove and/or refine any of this information. Note that we do not pre-populate DDP with requirements. Generally, these are highly application specific. Hence, we leave it to users to provide the requirements, and to provide the estimates of how much each risk impacts each requirement (the "impact" values in DDP).

As an alternative to introducing requirements and impacts, users may instead *directly* assess the severity of each risk. The advantage of this direct approach is that it is a lot less effort. The disadvantages are that it precludes the option to trade requirements, and it is a more prone to flaws of subjectivity than going through the more disciplined process of linking risks to requirements.

3.3 Connections to other tools and approaches

3.3.1 Estimation and Planning - Ask Pete

Ask Pete is a NASA developed tool to do estimation and planning of software assurance activities. It uses COCOMO II to help in its estimation of cost and schedule of the development project at hand, and combines this with information of NASA Glenn policies for software assurance, and NASA IV&V criteria for rating the criticality of projects. Ask Pete generates project plans for the software assurance activities appropriate for the project.

Ask Pete and ARRT have been made to communicate information back and forth [4]. Briefly, Ask Pete is run to generate a recommended set of assurance activities. These are then transferred into ARRT where they can be scrutinized, and tailored if so desired. The results of tailoring are then transferred back to Ask Pete for inclusion in the project plans it generates. In essence, this combination blends the strengths of Ask Pete at estimation, planning and documentation generation with the strengths of DDP at value maximization by balancing the costs (of assurance activities) against their values (of risk reduction, and therefore increased requirements attainment).

3.3.2 Optimization - TAR2

Value maximization in DDP is attained by judicious choice of the mitigations to perform. In DDP applications to date we have relied upon the experts to make this choice, guided by DDP's automatic calculations and cogent visualizations. A more automated approach to value maximization is desirable. Value maximization in DDP is, in essence, a traditional optimization problem in a domain where choices are binary (whether or not to perform a mitigation).

To solve this problem, we have explored the use of the machine-learning based approach of Tim Menzies [5], in the form of his TAR2 "treatment learner". Briefly, TAR2 is able to identify the key decisions – which mitigations to perform, and which mitigations to *not* perform – so as to converge upon a value maximizing solution. TAR2 thus has the desirable characteristic of locating near-optimal solutions, and doing so in such a way that identifies the most critical decisions to make to move towards those solutions. TAR2's identification of which decisions are the most critical is of especial value. It focuses the human experts on the decisions that matter the most, out of an otherwise dauntingly large number of alternatives. Furthermore, it offers them the opportunity to inject additional knowledge into the value maximization process. For example, once prompted by TAR2 with a candidate solution, the experts can indicate that some of the combinations of mitigations are infeasible and have TAR2 then search for alternative solutions that avoid these infeasible combinations. For more details, see [3].

We have also done some experiments using genetic algorithms to automate the value maximization process, yielding promising results. They tend to be faster than using Menzies' TAR2 (because TAR2 requires the generation of a large number of samples), but lack the ability to identify which of the decisions in a recommended solution are the most crucial. Overall, we think that a blend of automation and expert involvement will continue to be needed in value maximization, and believe this area to be worthy of further attention.

3.3.3 Further analysis

We are interested in building links between DDP and other analysis models. We believe DDP's strength is in the early phases of developments, when detailed designs are lacking. The use of DDP allows experts to home in on the major problem areas of their development, and suggest what to do about them (which mitigations to apply and/or which requirements to discard). As these decisions are made, the design begins to emerge. This is the point at which other analysis models (e.g., sophisticated cost estimation models, probabilistic assessment tools) can and should be brought to bear on what are now known to be the most critical aspects of the overall problem. At the very least, we would like to be able to transfer knowledge from DDP to these tools, giving them a head start on building their more detailed models. More ambitiously, we think there will be more synergistic combinations, in which information flows back and forth to allow detailed analysis results to return to DDP, guide the exploration of alternatives, lead to further detailed analyses, etc. We have only just started to pursue connections such as these.

4. ACKNOWLEDGEMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology. Contributions from, and discussions with, Burton Sigal (JPL), Patrick Hutchinson (Wofford College, Spartanburg SC), Peter In (Texas A&M), John Kelly (JPL), Tim Kurtz (NASA Glenn), James Kiper (Miami Univ., Ohio) and Tim Menzies (U. British Columbia) have been most useful.

5. REFERENCES

[1] S.L. Cornford, M.S. Feather & K.A. Hicks. "DDP – A tool for life-cycle risk management", IEEE *Aerospace Conference*, Big Sky, Montana, Mar 2001, pp. 441-451.

[2] M.S. Feather, B. Sigal, S.L. Cornford & P. Hutchinson. "Incorporating Cost-Benefit Analyses into Software Assurance Planning", to appear in Proceedings, 26th IEEE/NASA Software Engineering Workshop, Greenbelt, Maryland November 27-29 2001.

[3] M.S. Feather & T. Menzies. "Converging on the Optimal Attainment of Requirements", in submission.

[4] M.S. Feather & T. Kurtz. "Putting it All Together: Software Planning, Estimating and Assessment for a Successful Project", in Proc. of 4th International Software & Internet Quality Week Conference, Brussels, Belgium, Nov 2000.

[5] T. Menzies & Y. Hu. "Constraining Discussions in Requirements Engineering via Models", *1st International Workshop on Model Based Requirements Engineering*, San Diego, California, Dec 2